Public Leaderboards Are Not Deployment Tests: A Reproducible CPU-Only Robustness Audit for BirdCLEF+ 2026

Mathieu Weill[1]

[1]*Independent participant, France*

Abstract

BirdCLEF+ 2026 was framed as a large-scale acoustic species identification challenge, but the practical difficulty went beyond classification accuracy. Final submissions had to run as offline Kaggle notebooks, on CPU, within a 90-minute runtime limit, while predicting hundreds of species from continuous passive acoustic monitoring recordings. This setting made the competition not only a modeling benchmark, but also a deployment benchmark.

This working note presents a reproducible CPU-only inference pipeline and a robustness audit of high-performing public bioacoustic solutions. The pipeline combines Perch acoustic embeddings, ProtoSSM-style temporal modeling, residual correction, class-wise MLP probes, distilled sound event detection, rank-based blending, rare-class post-processing, and taxonomic smoothing. The engineering work focused on making these components run reliably under official code-competition constraints: offline dependency packaging, ONNX acceleration, input-source verification, defensive notebook execution, and strict submission validation.

The best public leaderboard score obtained was 0.950. The selected final submission scored 0.942 on the private leaderboard, ranking 502nd out of 4244 teams. Rather than treating this public–private drop as a failure only, this note uses it as evidence for a broader lesson: public leaderboard performance is not a deployment guarantee. Several apparently different public pipelines plateaued around 0.949–0.950 public score, suggesting strong correlation and limited robustness to hidden-test distribution shift. To make this observation useful beyond one competition, I introduce CRISP-Bioacoustic, a lightweight audit protocol for CPU-constrained reproducible inference pipelines. The main contribution of this work is therefore a deployment-oriented view of bioacoustic competition systems: strong public ensembles are useful, but robust conservation AI requires reproducibility, independent validation, and careful resistance to public leaderboard overfitting.

Keywords

BirdCLEF+ 2026, passive acoustic monitoring, bioacoustics, reproducibility, CPU inference, leaderboard robustness, AI-assisted engineering

1. Introduction

Passive acoustic monitoring is one of the most promising tools for biodiversity assessment at scale. In ecosystems such as the Pantanal wetlands, continuous recorders can collect long-term soundscape data across seasons, habitats, and recording conditions. The scientific bottleneck is no longer simply gathering audio. The bottleneck is converting large volumes of noisy, field-recorded audio into reliable species-level information.

BirdCLEF+ 2026 addressed this problem by asking participants to identify wildlife species from five-second windows of continuous recordings in the LifeCLEF evaluation setting [1, 2]. The task was multi-label and highly imbalanced: each row could contain several species, and many target classes had few or no positive examples in the labeled data available for training. The evaluation metric was a macro-averaged ROC-AUC variant that ignores classes without positive labels in the evaluation split. This made ranking quality and rare-class behavior central to the final score.

A second difficulty was operational. BirdCLEF+ 2026 was a Kaggle code competition. Submissions had to run as notebooks with internet disabled. CPU notebooks were limited to 90 minutes. GPU submissions were not practically usable for final inference because the GPU runtime allowance was too

CLEF 2026 Working Notes, September 21–24, 2026, Jena, Germany
mathieuweill@gmail.com (M. Weill)

short. A solution therefore had to be accurate, but also reproducible, offline, CPU-efficient, and robust to Kaggle's execution environment.

This distinction matters. Many public notebooks reached strong public leaderboard scores, but reproducing and submitting them was not automatic. Some depended on exact notebook outputs, manually attached Kaggle inputs, offline wheels, hidden execution state, or public leaderboard-tuned blending weights. In a research context, these are reproducibility issues. In a deployment context, they are reliability issues.

The work described here started as an attempt to build a competitive BirdCLEF+ 2026 submission. It became a practical audit of what makes a bioacoustic inference pipeline reproducible and robust under deployment-like constraints. The final result was not a medal-winning leaderboard placement. Nevertheless, it exposed a useful and general lesson: public leaderboard optimization can produce systems that appear strong but are not necessarily robust on the private split. The value of this note is therefore not a claim of state-of-the-art accuracy, but a reproducible account of how strong public bioacoustic systems behave under CPU-only deployment constraints, and how public leaderboard performance can diverge from private robustness.

The contributions of this working note are: a reproducible CPU-only inference stack combining Perch embeddings, temporal sequence modeling, residual correction, distilled SED models, rank blending, and taxonomic post-processing; an engineering account of the failure modes that prevent strong public bioacoustic notebooks from running reliably under official submission constraints; an empirical public–private robustness analysis showing a public plateau around 0.949–0.950 and a private drop to 0.942 for the selected submission; CRISP-Bioacoustic, a reusable audit protocol for checking CPU-constrained bioacoustic inference pipelines before trusting public leaderboard performance; and documentation of an AI-assisted engineering workflow used to diagnose logs, patch notebooks, organize the repository, and prepare this note under time pressure.

The central claim is not that this was the most accurate BirdCLEF+ 2026 system. The central claim is that public score, reproducibility, and deployability are different properties, and that future bioacoustic systems should be evaluated with all three in mind.

2. Competition Setting

For each five-second audio window, participants had to predict one probability per target species. The submission file had to match `sample_submission.csv`, with one `row_id` column and one column for each species. The final evaluation used a macro-averaged ROC-AUC variant that skipped classes with no positive labels.

This metric has two practical consequences. First, rare species matter: each evaluated class contributes to the macro average. Second, probability ranking matters more than a fixed threshold. A model can have poorly calibrated probabilities and still perform well if its ranking within each species is strong. This motivated the use of rank-based blending and class-specific post-processing.

In the working pipeline, the trusted full-window subset contained 59 soundscape files, corresponding to 708 five-second windows. The target space contained 234 classes, but only 71 were active in these trusted windows. This imbalance made it difficult to rely on supervised training alone. The solution therefore depended heavily on pretrained acoustic representations and external public model artifacts.

The official submission constraints were equally important: internet access was disabled; CPU notebook runtime was limited to 90 minutes; the final output file had to be named `submission.csv`; all dependencies and model files had to be attached as Kaggle inputs; and GPU submissions were not practically usable for normal inference. In this setting, runtime and reproducibility were part of the method, not implementation details.

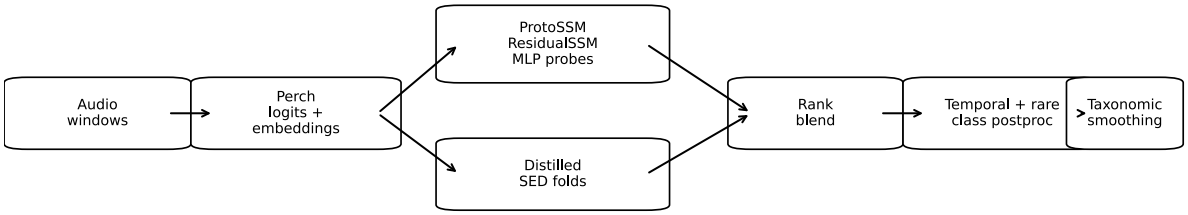

Figure 1: Overview of the CPU-only inference stack. Raw audio is converted to Perch logits and embeddings. The Perch features feed temporal sequence models, MLP probes, and residual correction. A separate distilled SED branch provides complementary event-level predictions. Outputs are combined with rank blending and refined with temporal, rare-class, and taxonomic post-processing.

3. Method Overview

The final pipeline family combined two broad sources of signal. The first was a Perch-based sequence-modeling stack. The second was a distilled sound event detection stack. The outputs were merged with rank-based blending and then refined using post-processing.

The main stages were: locate competition files and attached Kaggle inputs; install or import offline dependencies; load Perch or Perch-ONNX features; construct window-level and file-level tensors; run ProtoSSM-style sequence inference; apply residual correction and MLP probe refinements; run distilled SED ONNX models; blend sequence and SED predictions; apply temporal, taxonomic, and rare-class post-processing; validate and write `submission.csv`.

The design was intentionally modular. Under CPU-only constraints, a single large end-to-end model would have been difficult to train and run reliably. A modular stack allowed pretrained representations, public model artifacts, and lightweight corrections to be combined while keeping the notebook within the runtime limit.

4. Perch Embeddings and CPU Inference

Perch was used as the primary acoustic representation [3]. It provided both class logits and 1536-dimensional embeddings. These embeddings were strong enough to support downstream classifiers and sequence models despite limited labeled training data.

Perch features were used in four ways: as direct class-level evidence for mapped species; as high-dimensional input to ProtoSSM-style temporal models; as PCA-reduced features for MLP probes; and as cached arrays to reduce repeated inference cost.

A practical challenge was that TensorFlow-based Perch inference on CPU could be slow. Some notebook variants required TensorFlow versions that were not directly available in the Kaggle environment. Offline TensorFlow and TensorBoard wheels therefore had to be attached as Kaggle datasets and installed during execution. Where possible, ONNX versions of Perch were used instead. ONNX inference significantly reduced runtime and simplified CPU execution.

The pipeline searched `/kaggle/input` for key files such as `perch_v2.onnx`, `perch_v2_no_dft.onnx`, `full_perch_meta.parquet`, and `full_perch_arrays.npz`. When cached Perch arrays were available, the notebook used them. When they were missing, it recomputed features and saved cache files to `/kaggle/working`. This behavior helped development, but also showed why exact input-source documentation is necessary for reproducibility.

5. Temporal Modeling and Lightweight Corrections

The sequence component used ProtoSSM-style temporal models. The goal was to model context across adjacent five-second windows from the same soundscape. Wildlife vocalizations are temporally structured; a species detected in one window may also influence the likelihood of nearby windows.

A representative configuration used a 1536-dimensional embedding input, a model dimension of 256, state dimension 16, three SSM layers, dropout 0.15, cross-attention with four heads, and 234 output classes. The data was reshaped from individual windows into file-level sequences. In the trusted full-window subset, this corresponded roughly to 59 files by 12 windows by 1536 embedding dimensions.

In final submission mode, full training was usually skipped and pretrained weights were loaded from attached Kaggle datasets. This choice was necessary for runtime stability. The model acted as an inference-time temporal aggregator and scorer rather than as a fully retrained competition-specific architecture.

Two lightweight correction mechanisms were used. The first was a ResidualSSM model. Instead of predicting class probabilities from scratch, it learned adjustments to base scores. This made it computationally cheaper and less likely to destroy useful structure from the base model. Correction weights between 0.10 and 0.40 were tested in some variants. Internal OOF diagnostics often showed only small differences between values around 0.25–0.35, which was treated as a warning that local validation was not strong enough to finely tune this parameter.

The second mechanism was a set of MLP probes trained on PCA-reduced Perch embeddings. Depending on the variant, embeddings were reduced to 64 or 128 dimensions. With 128 dimensions, around 90% of embedding variance was retained; with 64 dimensions, around 81% was retained. A typical run trained approximately 58 class-specific probes. The probes were useful for classes with enough positive examples, but their value for rare classes was uncertain.

6. Distilled SED Models and Post-processing

The second main model family was a distilled sound event detection system. This component used ONNX fold models, typically `sed_fold0.onnx` through `sed_fold4.onnx`.

The SED models provided complementary predictions. While the Perch/ProtoSSM stack relied heavily on pretrained embeddings and temporal sequence modeling, the SED stack behaved more like an event detector. Blending these two families improved the public score. A common internal blend used approximately 60% ProtoSSM-style predictions and 40% distilled SED predictions.

The SED component also exposed one of the clearest reproducibility hazards. If the Kaggle dataset containing the SED folds was not attached, the notebook failed even though all Python code was valid. The fix was not algorithmic; it was an input-source audit. This became one motivation for formalizing the CRISP-Bioacoustic protocol described later.

Raw model scores came from different distributions. Some were logits, some were calibrated probabilities, and some were post-processed outputs. Direct averaging could therefore be misleading. Rank-based blending was used to reduce sensitivity to calibration differences.

The final public-performing systems used combinations of rank blending between ProtoSSM and SED predictions; per-taxon temperature scaling; file-level confidence scaling; temporal smoothing across neighboring windows; rank-aware scaling; adaptive rare-species processing; sonotype mirroring for selected columns; and taxonomic smoothing. Rare-class post-processing was especially important because of macro-AUC, although highly tuned class-specific thresholds were risky in submission mode when validation was weak.

Operational definitions. Rank-aware scaling denotes a monotonic transformation of class probabilities or ranks to emphasize high-confidence predictions while preserving ordering. Taxonomic smoothing shares or smooths confidence among related species or taxonomic groups to reduce inconsistent predictions. Sonotype mirroring is a heuristic copying or coupling of selected acoustically

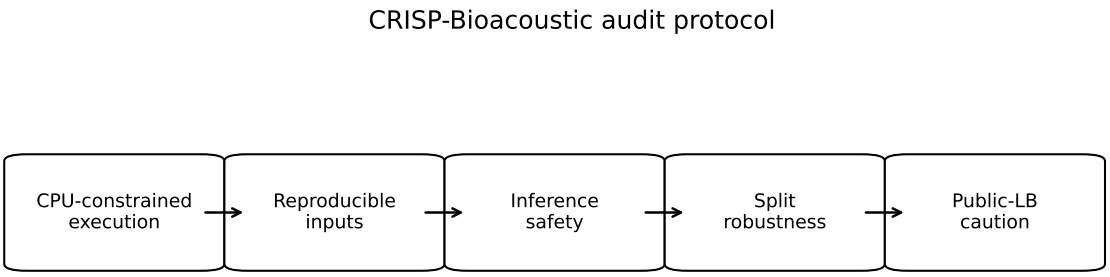

Figure 2: CRISP-Bioacoustic audit checks. The protocol is designed to catch practical reproducibility failures before they become failed submissions or misleading public leaderboard results.

Table 1
CRISP-Bioacoustic audit checks and observed failures.

Audit check	Observed failure	Repair
Dependency audit	TensorFlow or ONNX Runtime unavailable offline	Attach compatible wheel datasets
Input-source audit	SED folds or Perch cache missing	Search `/kaggle/input`; attach missing datasets
Shape audit	Dry-run output had only a few rows	Validate against `sample_submission.csv`
Execution-order audit	`site_to_idx`, `probe_models`, `_runSED_once` undefined	Add defensive initialization
Component audit	Multiple variants plateaued at 0.949–0.950	Treat as correlated models
Public-private audit	Public 0.950 selected model became private 0.942	Avoid trusting public score alone

or taxonomically similar output columns when one model family provides stronger signal. Adaptive rare-species processing is a class-specific adjustment for rare targets whose validation positives are too sparse for standard calibration. The xSED rank blend is the rank-based blend between ProtoSSM-style predictions and distilled SED predictions, often around 0.60/0.40 in this work.

7. CRISP-Bioacoustic Audit Protocol

The main methodological contribution of this note is CRISP-Bioacoustic, a lightweight audit protocol for CPU-constrained bioacoustic inference systems. CRISP stands for CPU-constrained execution, Reproducible inputs, Inference safety, Split robustness, and Public-leaderboard caution.

The protocol consists of six checks. The dependency audit asks whether all packages can be installed or imported without internet. The input-source audit verifies that every required model, cache, and auxiliary file is visible under `/kaggle/input`. The shape and submission audit validates row count, column order, missing values, numeric ranges, and alignment with `sample_submission.csv`. The execution-order audit checks whether submit-mode variables are initialized defensively. The component-independence audit asks whether new variants add independent signal or merely reproduce the same public leaderboard behavior. The public-private risk audit asks whether a change is supported by local validation or independent reasoning, rather than only by public leaderboard feedback.

The protocol is simple, but intentionally practical. Its output is a documented pass/fail record for each check, helping future participants and researchers distinguish a leaderboard-working notebook

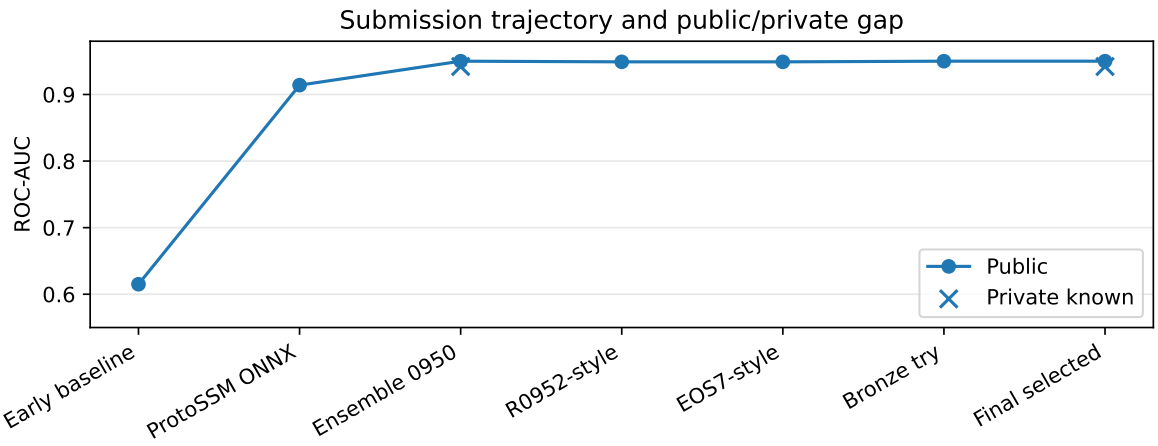

Figure 3: Main score trajectory. The largest improvement came from moving from a weak baseline to Perch/ProtoSSM/ONNX inference and then adding the public ensemble/SED stack. The final private score was below the best public score.

Table 2
Submission trajectory.

Run family	Local/OOF diagnostic	Public LB	Private LB	Status	Notes
Early baseline	not used for final selection	0.615	–	exploratory	Format and execution check
ProtoSSM ONNX patched	diagnostic only	0.914	–	submitted variant	First competitive CPU/offline pipeline
Ensemble 0950 pipeline	diagnostic only	0.950	–	public family	Best stable public family
R0952-style submission	not used for final selection	0.949	–	reproduced public family	Close public notebook family
EOS7-style submission	not used for final selection	0.949	–	reproduced public family	Alternative public family
Bronze/high-risk try	not used for final selection	0.950	–	high-risk try	More complex public blend
Final selected submission	diagnostic only	0.950 family	0.942	selected final	Final rank 502/4244

from a deployable inference pipeline.

8. Experiments and Leaderboard Behavior

The development process produced several submission families. Table 2 summarizes the most important ones.

The first major improvement came from the Perch/ProtoSSM/ONNX pipeline, which moved from a weak baseline to 0.914 public score. The next jump came from integrating distilled SED predictions and public ensemble post-processing, reaching 0.950 public.

After that, progress stalled. Several variants that looked different in code or notebook source still scored 0.949–0.950. This included R0952-style and EOS7-style variants, as well as a high-risk blend intended to move beyond the plateau. The plateau suggested that the variants were not independent enough.

The final selected submission had a best public score of 0.950, a private score of 0.942, and a final rank of 502 out of 4244 teams. This result was disappointing from a competition standpoint, but informative from a robustness standpoint. The public leaderboard represented only part of the test distribution. The final private leaderboard used the remaining hidden portion. A system that performs well on the public split may therefore fail to generalize.

The observed public plateau around 0.949–0.950 had three implications. First, many teams likely

used similar public components. Perch embeddings, ProtoSSM-style models, distilled SED outputs, and taxonomic post-processing appeared repeatedly in high-scoring public notebooks. These components are strong, but their errors may be correlated. Second, small changes to blend weights did not necessarily add new information. A blend may change the public score slightly without improving private robustness. Third, local validation was too weak to confidently choose among many near-identical public variants. The private score of 0.942 suggests that public leaderboard optimization had become partly disconnected from deployment robustness.

8.1. Sensitivity observations

The observed sensitivity evidence was practical rather than a controlled ablation study. Changing blend weights among strongly correlated public families often did not move the public score beyond 0.949–0.950. Missing SED folds caused hard failure, making input availability more critical than algorithmic tuning. Residual correction weights around 0.25–0.35 showed similar OOF diagnostics in observed logs, indicating weak discrimination by local validation. Rank blending and taxonomic smoothing changed the output distribution, but public score gains were not reliable evidence of private robustness. The final public plateau and private drop therefore suggest a concrete public-leaderboard tuning risk.

8.2. Artifact and execution checklist

The audit covered the following reproducibility artifacts. Kaggle notebooks/kernels used or reproduced were birdclef2026-protossm-onnx-submit, birdclef2026-ensemble-0950-submit, birdclef2026-r0952-submit, birdclef2026-eos7-submit, and birdclef2026-bronze-try1. The main repository folders were notebooks/original/, notebooks/generated/, notebooks/patched/, kaggle_upload/, and docs/. Required Kaggle inputs included mathieuw/birdclef2026-tf220-wheels, rishikeshjani/perch-onnx-for-birdclef-2026, tuckerarrants/perch-v2-no-dft-onnx, hideyukizushi/sgkfk-202604041716, jaejohn/perch-meta, tuckerarrants/bc2026-distilled-sed-public, and, when referenced by the final notebook, tuckerarrants/birdclef-2026-waveform-cache. Key files were perch_v2.onnx, perch_v2_no_dft.onnx, proto_ssm_best.pt, residual_ssm_best.pt, sed_fold0.onnx–sed_fold4.onnx, full_perch_arrays.npz, and full_perch_meta.parquet. The observed runtime setting was CPU-only Kaggle inference with internet disabled, ONNX Runtime 1.24.4, TensorFlow 2.19/2.20 depending on notebook path, PyTorch 2.10 CPU, and a 90-minute CPU limit. The output was submission.csv, validated against sample_submission.csv.

9. AI-assisted Engineering Workflow

AI-assisted tools were used throughout the project, but not as autonomous experimenters. Their main role was to turn noisy Kaggle logs and repository traces into actionable debugging steps: identifying missing inputs, suggesting defensive initialization, drafting small patch scripts, and organizing manuscript notes. Several failures were reproducibility failures rather than modeling failures, including missing SED folds, broken local environments, invalid notebook sources, skipped-cell variables, and dry-run shape mismatches. Every patch was manually inspected, every reported score was taken from Kaggle outputs, and the scientific interpretation remains the author's own.

10. Discussion and Limitations

The most important modeling lesson is that pretrained bioacoustic representations are extremely valuable. Perch embeddings provided a strong foundation that allowed competitive systems to be built with limited labeled data. Without such representations, the early baseline was far from competitive.

The most important engineering lesson is that reproducibility must be treated as part of the method. Offline wheels, input discovery, ONNX acceleration, and submission validation were not optional details. They determined whether the system could run at all.

The most important evaluation lesson is that public leaderboard performance can be misleading when many teams use correlated public pipelines. A dense cluster of teams at the same public score is a warning sign. It may indicate a strong shared baseline, but also limited independent signal.

This work has clear limitations. The system was an integration and audit of public and locally modified components, not a fully novel architecture developed from scratch. The public-private analysis is observational: I did not have access to private labels, so I cannot prove which exact classes or recording conditions caused the private drop. Local validation was weak because the trusted labeled subset was too small to reliably distinguish highly correlated 0.949–0.950 variants. Some components depended on public Kaggle artifacts; a fully archival release would require packaging all inputs into a single documented dataset. Finally, CRISP-Bioacoustic is a practical protocol, not a formal statistical guarantee. It reduces avoidable reproducibility and robustness failures, but it cannot replace strong validation data.

11. Conclusion

BirdCLEF+ 2026 showed that acoustic species identification is both a modeling problem and a deployment problem. The final pipeline combined Perch embeddings, ProtoSSM-style temporal modeling, residual correction, MLP probes, distilled SED models, rank blending, and taxonomic post-processing under strict CPU-only offline constraints. The best public score was 0.950, while the private score of the selected submission was 0.942, with a final rank of 502 out of 4244 teams.

The main contribution of this note is not a claim of state-of-the-art accuracy. It is a reproducibility and robustness perspective on high-performing bioacoustic inference pipelines. The proposed CRISP-Bioacoustic audit protocol formalizes the practical checks that were necessary to make public notebook components run as deployable submissions.

The broader lesson is simple: public leaderboards are useful debugging tools, but they are not deployment tests. For conservation applications, robust inference requires pretrained acoustic representations, efficient execution, careful validation, independent model diversity, and explicit resistance to public leaderboard overfitting.

Declaration on Generative AI

During preparation of this work, the author used ChatGPT, Codex-style coding assistance, and NotebookLM document synthesis tools for programming, debugging, log analysis, repository organization, grammar and spelling checks, and manuscript preparation. The author reviewed and manually verified all experiments, Kaggle outputs, claims, and scientific interpretations, and takes full responsibility for the content.

References

[1] L. Picek, L. Adam, S. Kahl, R. Bossy, L. Chrobak, H. Goëau, K. Papafitsoros, H. Klinck, W.-P. Vellinga, R. Planqué, T. Denton, K. Barnard, C. Nédellec, L. Deléger, M. Courtin, G. Martellucci, I. Moummad, F. Vinatier, P. Bonnet, A. Joly, Overview of LifeCLEF 2026: Ai challenges for biodiversity understanding and ecosystem management, in: International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF), Springer, 2026.

[2] S. Kahl, T. Denton, L. Sugai, L. Piatti, W. H. Arruda Moreno, M. M. Barbosa, M. Eibl, C. Z. Fieker, C. M. Garcia, D. L. Hokama Sousa, J. E. d. A. Júnior, R. C. Kridler, M. Lasseck, A. V. d. Melo, H. Müller, M. d. O. Neves, M. G. d. Reis, L. K. Sampaio Teles, K.-L. Schuchmann, J. L. M. M. Sugai, K. T. P. Azevedo, P. d. N. Lopes, M. I. Marques, H. Klinck, H. Glotin, H. Goëau, W.-P. Vellinga, R. Planqué, A. Joly, Overview of BirdCLEF+ 2026: Acoustic species identification in the Pantanal, South America, in: Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum, 2026.

[3] Google Research / Google DeepMind, Bird Vocalization Classifier / Perch pretrained acoustic model, 2026. Public model used as pretrained acoustic representation.